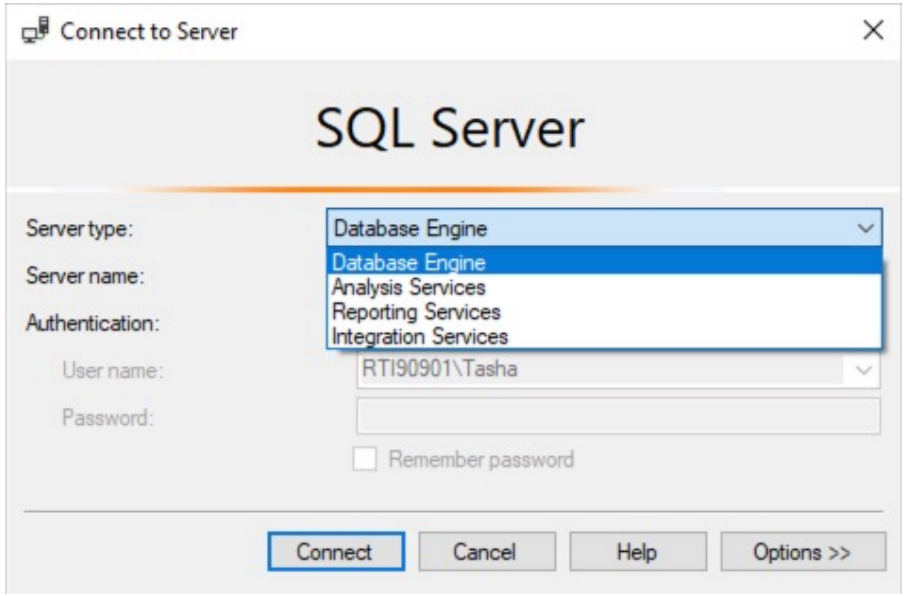
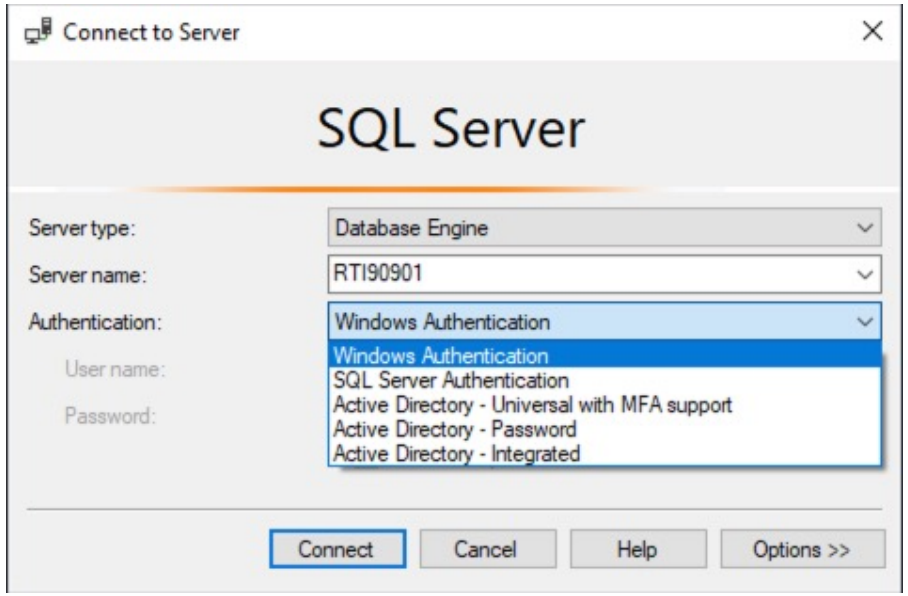
Microsoft SQL Server i Transact-SQL

SOFTVERSKI ALATI BAZA PODATAKA

SQL Server Management Studio - Konekcija ka serveru



Za Server type izaberemo Database Engine.



Za Authentication izaberemo Windows Authentication.

Klikom na Connect konektujemo se na ciljni server.

Rad sa bazom podataka

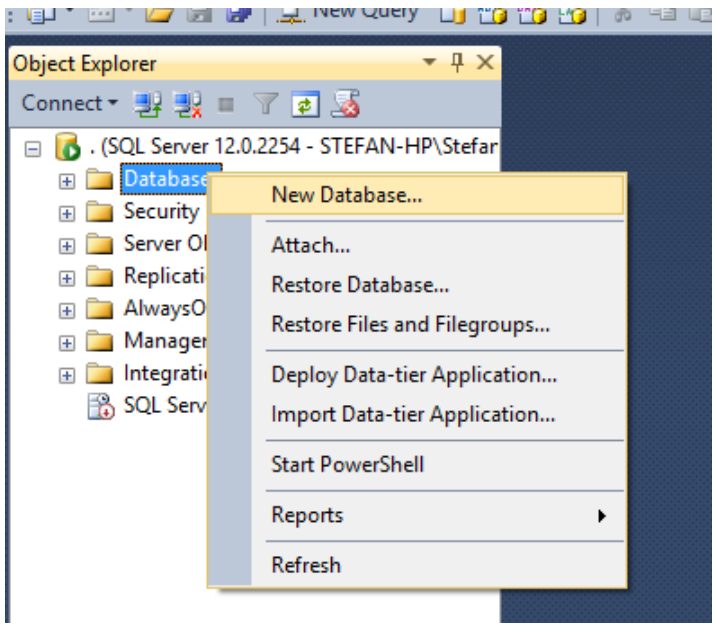
SQL Server baza može biti kreirana, izmenjena, ili obrisana:

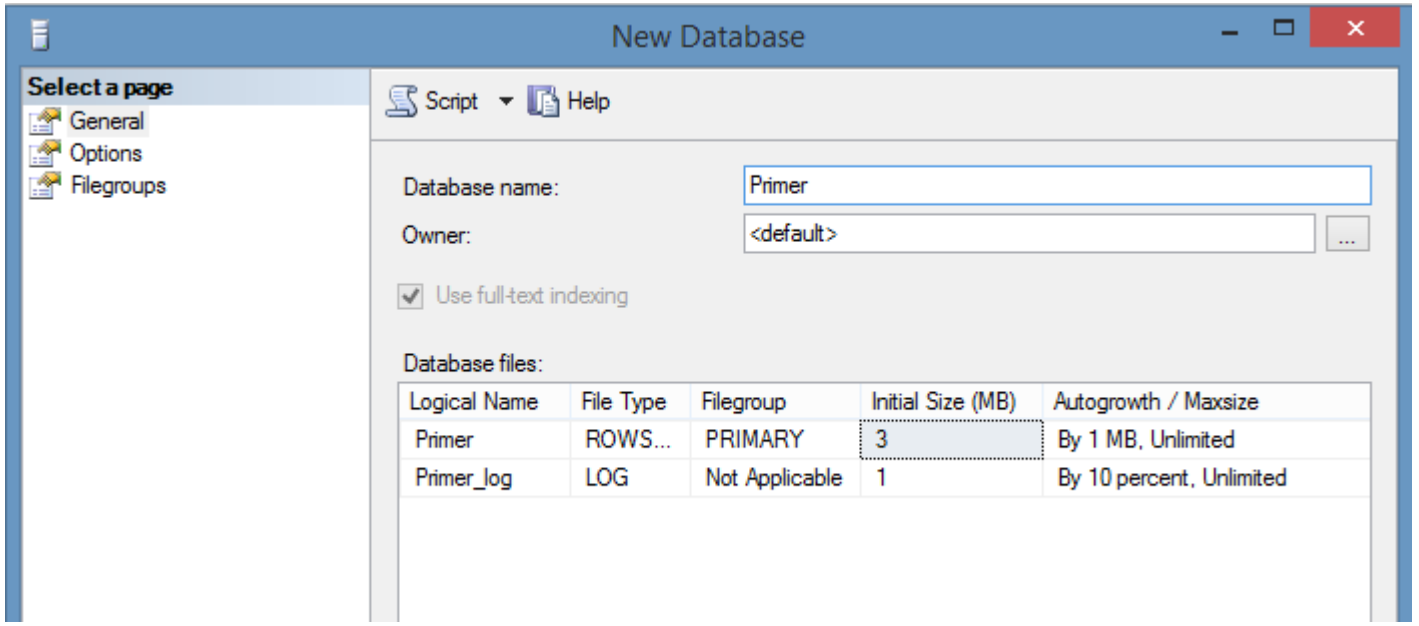
1. grafički – SQL Server Management Studio

2. korišćenjem upita

Rad sa bazom podataka

1. Kreiranje





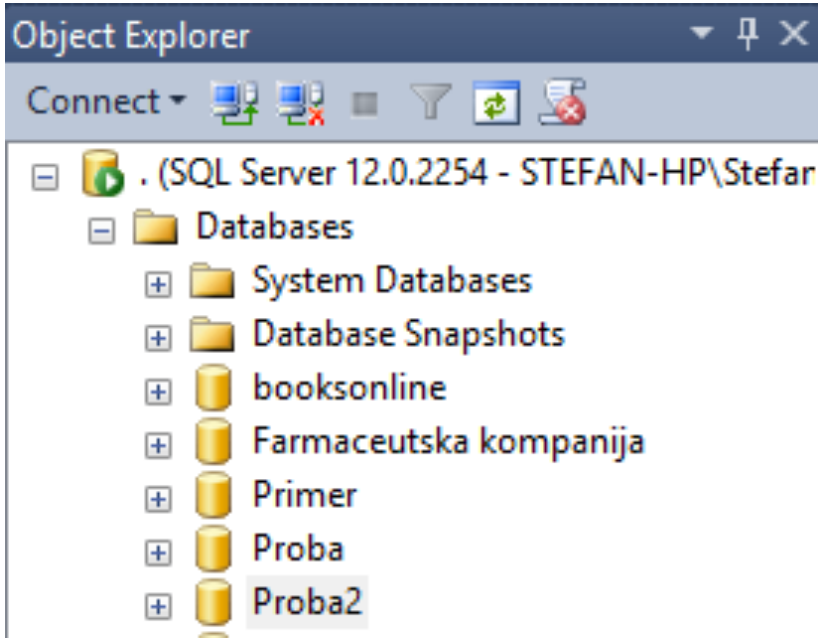
Desni klik na Database pa na New Database.

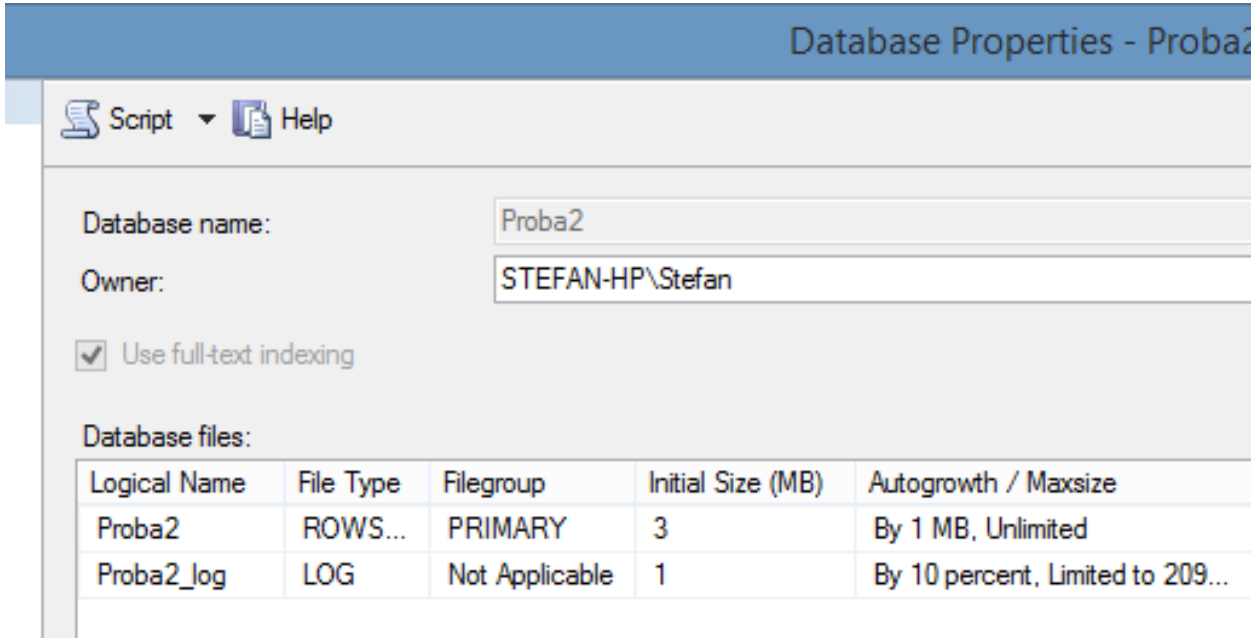
Unesemo Database name, u ovom slučaju Primer, i kliknemo na OK.

Sada je naša baza pod nazivom Primer kreirana.

Rad sa bazom podataka

```
Create database Proba2
```



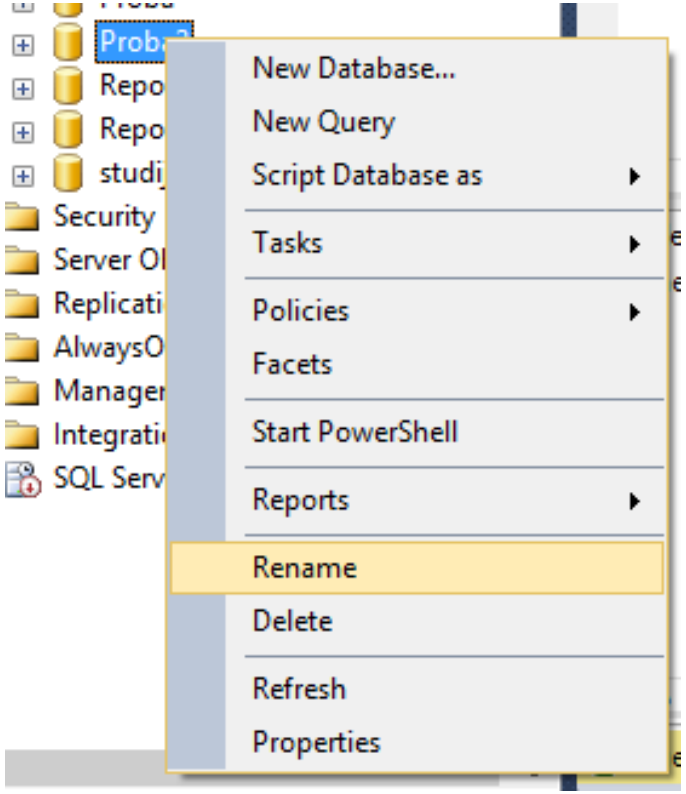


Kreirana je baza pod nazivom Proba2, i za nju su kreirana dva fajla:

- .mdf – Data File (sadrži same podatke)
- .ldf – Transaction Log File (koristi se za oporavak baze)

Rad sa bazom podataka

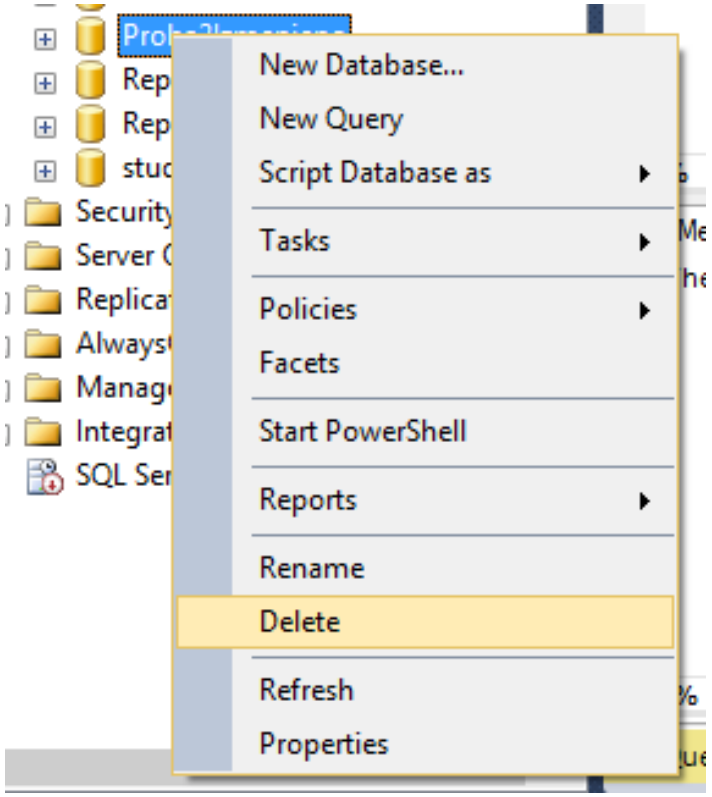
2. Izmena naziva




```
Alter database Proba2 Modify name = Proba2Izmenjeno

Execute sp_renamedb 'Proba2', 'Proba2Izmenjeno'
```

Rad sa bazom podataka

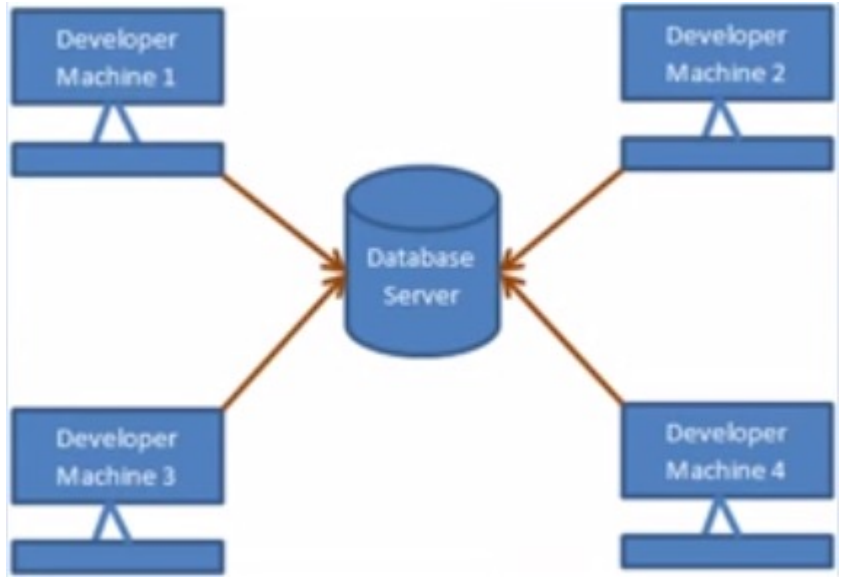
3. Brisanje




```
Drop database Proba2Izmenjeno
```

Brisanjem baze brišu se .mdf i .ldf fajl.

Rad sa bazom podataka



Ukoliko više korisnika pristupa istoj bazi na serveru, brisanje baze može biti problem.

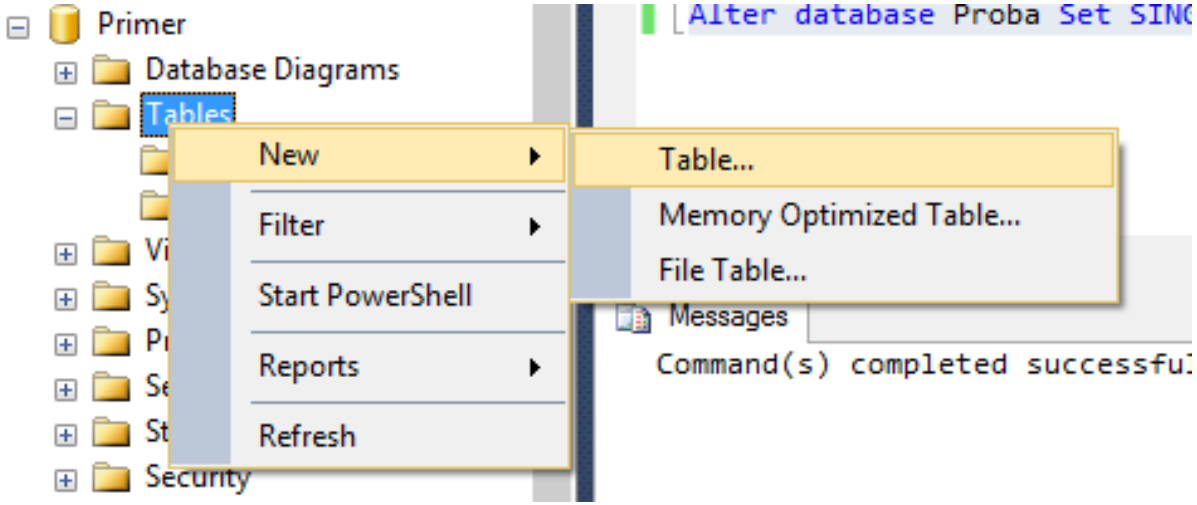
Ukoliko klijent na mašini 3 izvršava neki upit nad bazom i u toku tog izvršavanja klijent na mašini 2 pokuša da izbriše bazu neće uspeti i dobiće grešku:
Cannot drop database „DBName“ because it is currently in use.

Korisnik koji želi da izbriše bazu, a nad njom radi više korisnika treba da izvrši sledeću komandu:

```
Alter database Proba Set SINGLE_USER With RollBack Immediate
```

`RollBack Immediate` govori da transakcije svih ostalih korisnika treba odmah Rollback-ovati.

Kreiranje tabele



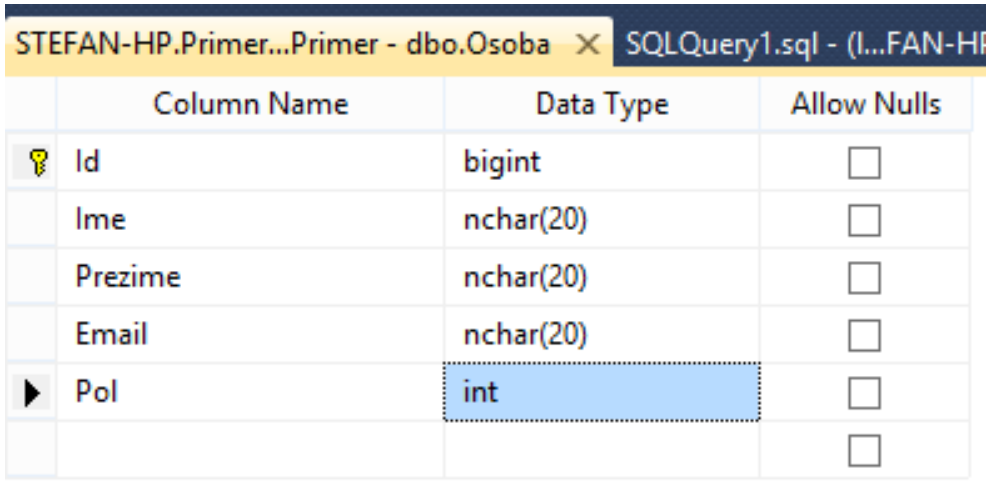


Tabela Osoba

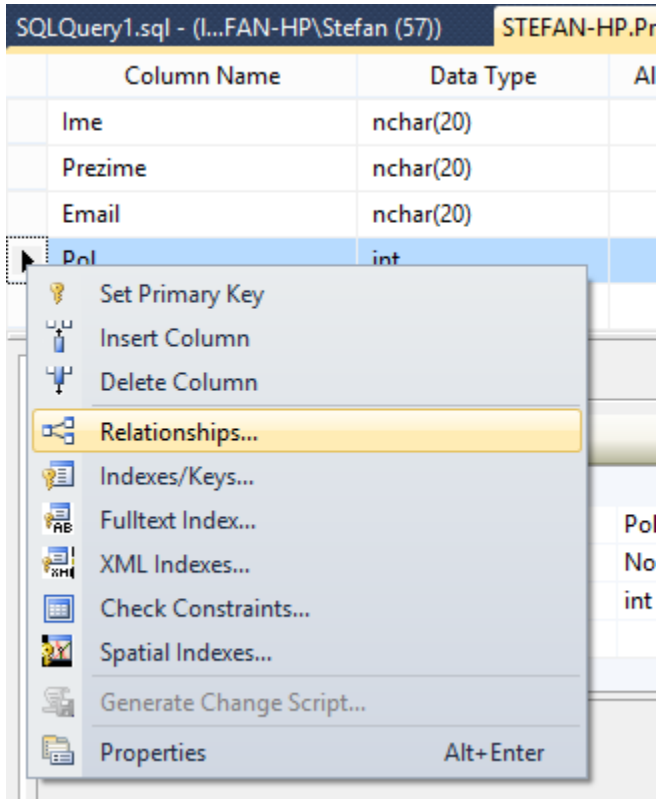
Kreiranje tabele

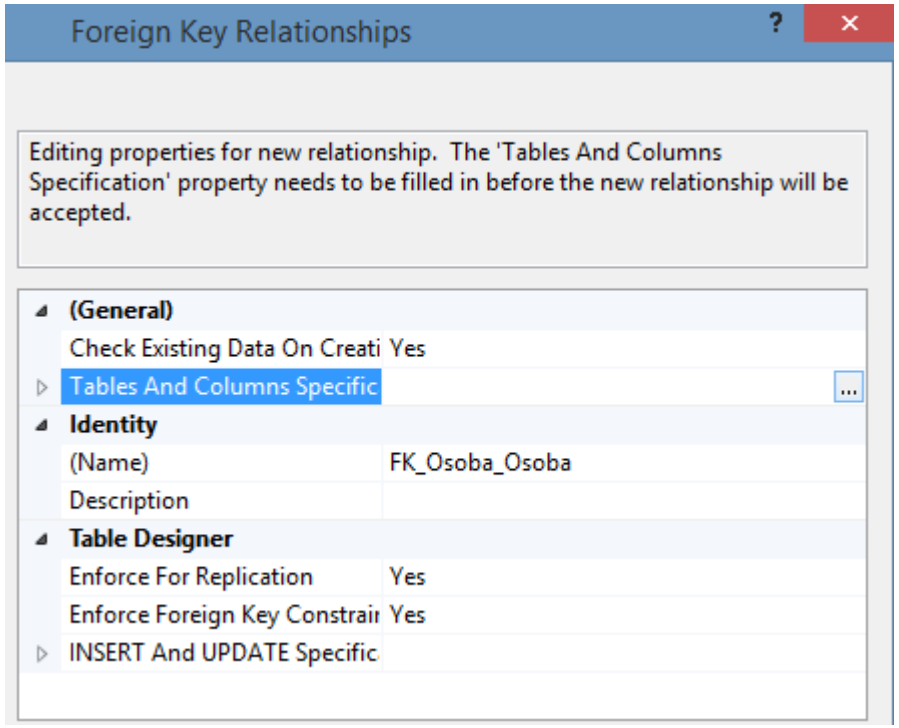
Kreiranje tabele Pol.

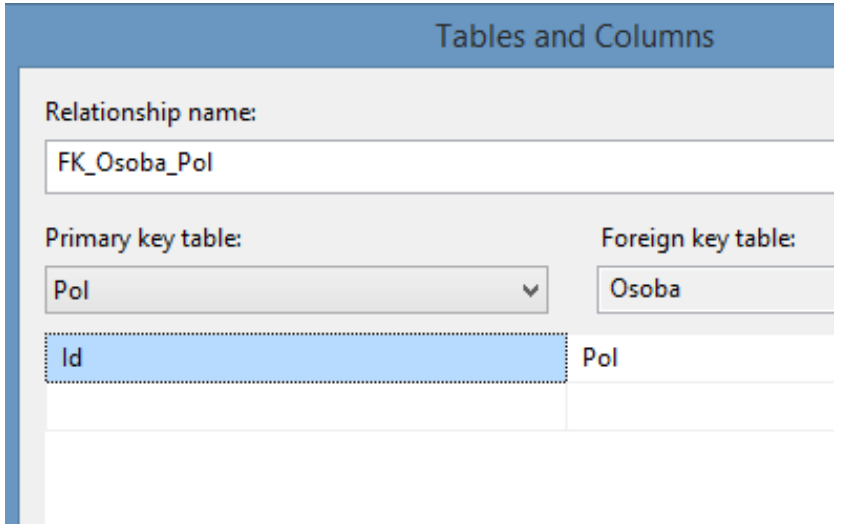
```
Use [Proba]
go
Create table Pol
(
Id int Primary Key,
Tip varchar(5) NOT NULL
)
```

`Use [Proba] go` – definiše bazu na koju će se upit odnositi

Strani ključevi

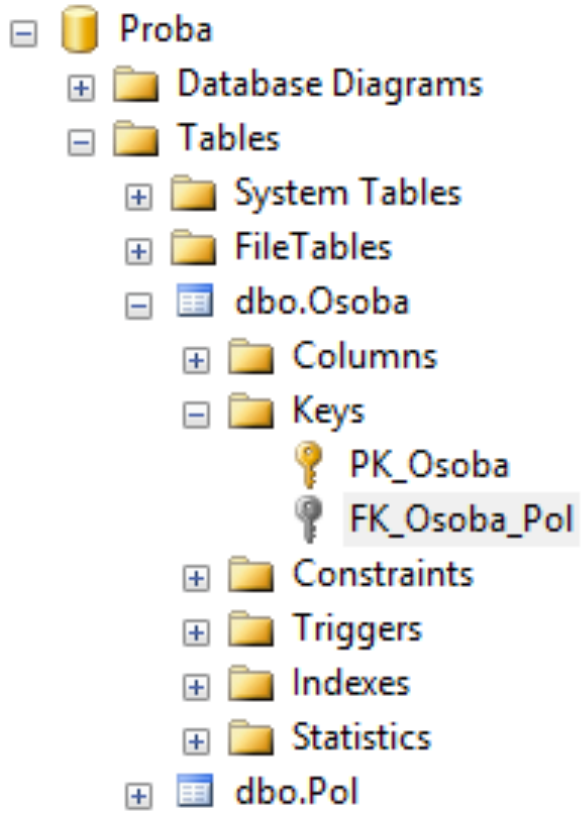






Strani ključevi

```
Alter table Osoba add constraint FK_Osoba_Pol
Foreign Key (Pol) references Pol(Id)
```

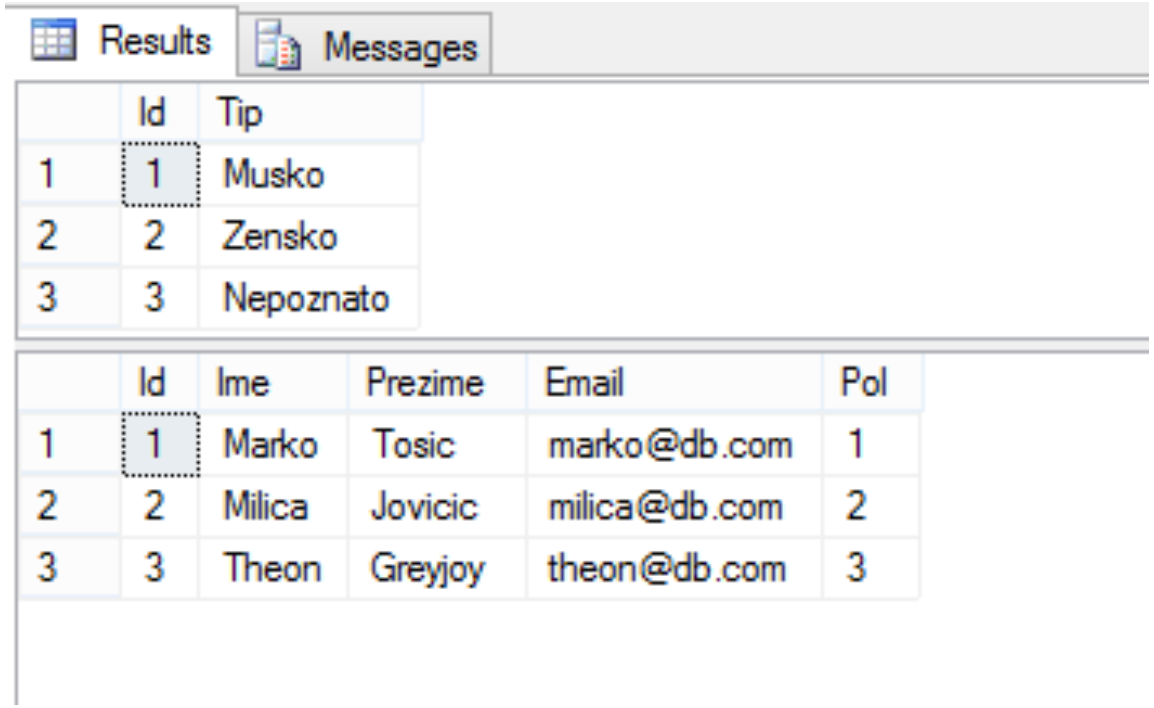


Default constraint

```
Insert into Pol (Id, Tip) Values (1,'Musko')
Insert into Pol (Id, Tip) Values (2,'Zensko')
Insert into Pol (Id, Tip) Values (3,'Nepoznato')

Alter table Osoba
Add constraint DF_Pol
Default 1 for Pol

Insert into Osoba (Id, Ime, Prezime, Email, Pol) values (1, 'Marko', 'Tosic', 'marko@db.com', 1)
Insert into Osoba (Id, Ime,Prezime, Email,  Pol) values (2, 'Milica', 'Jovicic', 'milica@db.com', 2)
Insert into Osoba (Id, Ime, Prezime, Email) values (3, 'Theon', 'Greyjoy', 'theon@db.com')
```

Default constraint

```
select * from Pol
select * from Osoba
```



	Id	Tip
1	1	Musko
2	2	Zensko
3	3	Nepoznato

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3

```
Alter table Osoba
Drop constraint DF_Pol
```

Referencijalni integriteti

Referencijalni integriteti:

1. No action – ne dozvoljava brisanje pa ne vrši nikakvu akciju (transakcija se rollback-uje)
2. Cascade – briše odgovarajuće redove u povezanim tabelama
3. Set Null – postavlja NULL vrednosti za odgovarajuće redove i kolone u povezanim tabelama
4. Set Default - postavlja Default vrednosti za odgovarajuće redove i kolone u povezanim tabelama

Referencijalni integriteti

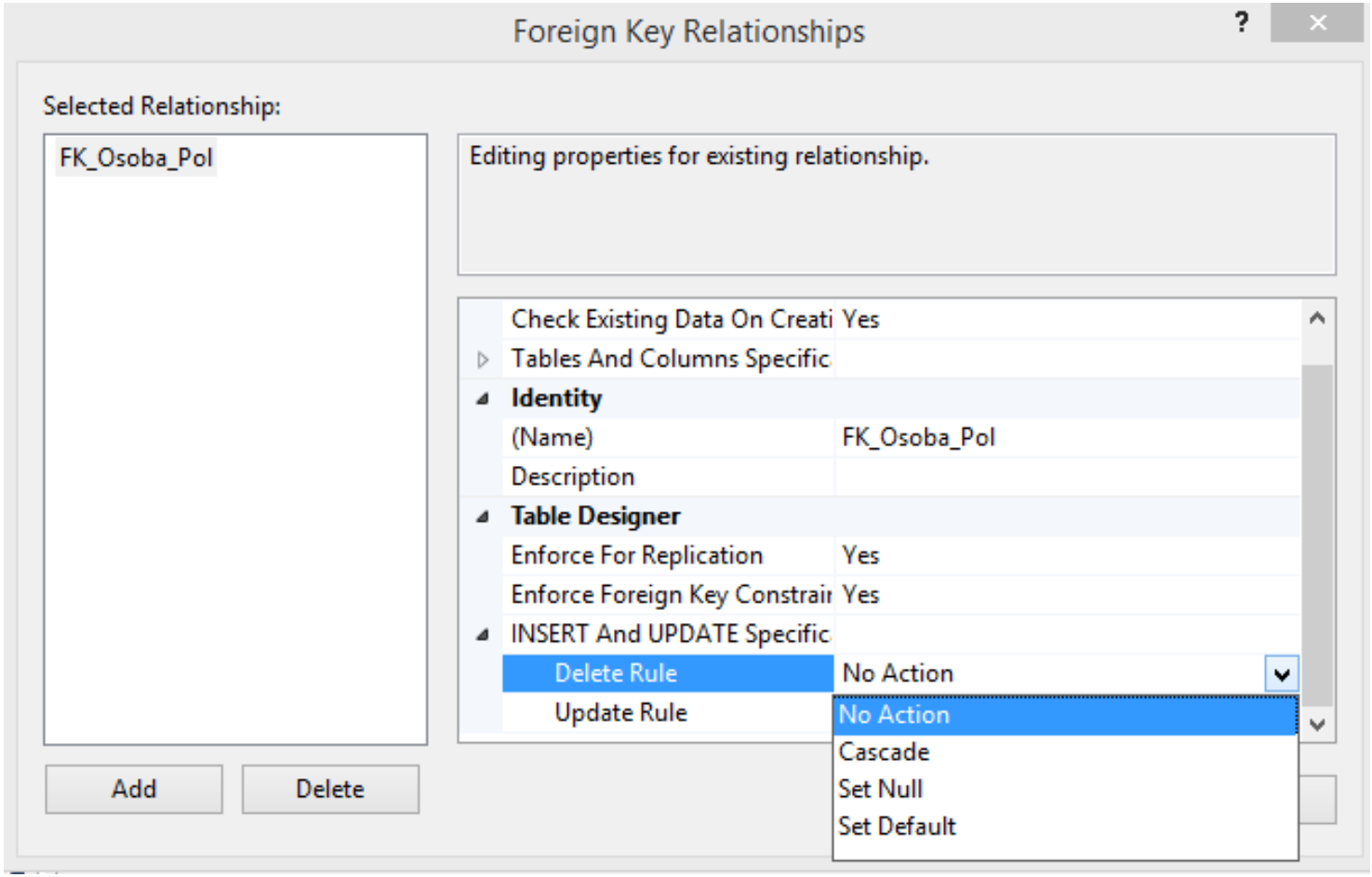
```
Insert into Pol(Id, Tip) Values(4, 'Vanzemaljac')

Insert into Osoba (Id, Ime, Prezime, Email, Pol)
values (4, 'Alien', '1', 'alien@db.com', 4)

Delete from Pol where Id = 4
```

```
The DELETE statement conflicted with the
REFERENCE constraint "FK_Osoba_Pol". The
conflict occurred in database "Proba", table
"dbo.Osoba", column 'Pol'.
```

Ovaj kod ne može biti izvršen jer je referencijalni integritet za Constraint „FK_Osoba_Pol“ automatski postavljen na „**NO ACTION**“

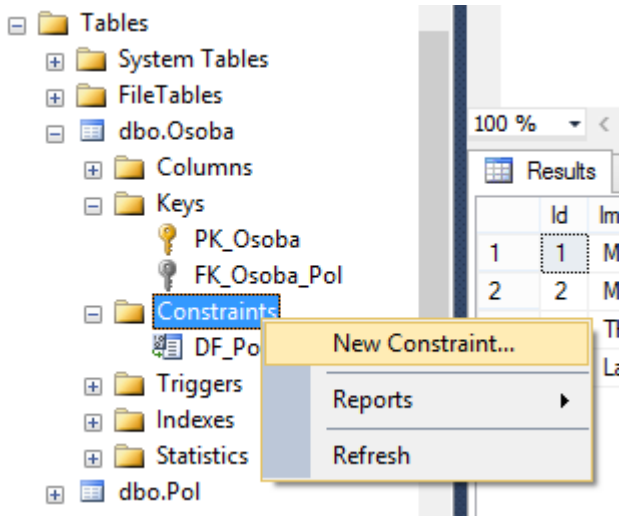
Referencijalni integriteti

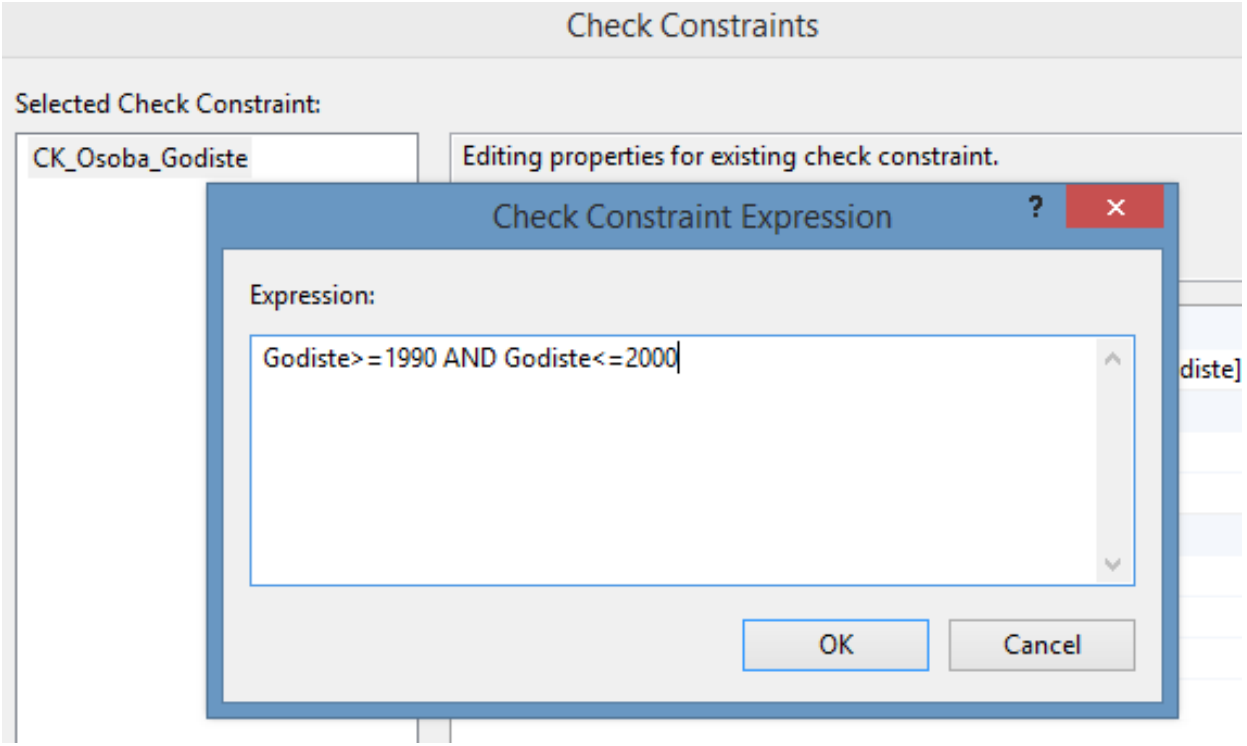
Iz tog razloga postavljamo drugačije referencijalne integritete.



Check constraints

```
Alter table Osoba
Add Godiste int
```





Check constraints

```
Insert into Osoba values (4, 'Lazar', 'Petrovic', 'lazar@db.com', 1, 2200)
```

```
The INSERT statement conflicted with the CHECK constraint "CK_Osoba_Godiste".
The conflict occurred in database "Proba", table "dbo.Osoba", column
'Godiste'.
```

Vrednost 2200 je van opsega, pa ovakav red ne možemo uneti u tabelu Osoba.

Check constraints

Column Name	Data Type	Allow Nulls
Prezime	nchar(20)	☐
Email	nchar(20)	☐
Pol	int	☐
Godiste	int	☑

```
Insert into Osoba values (4, 'Jovana', 'Peric', 'jovana@db.com', 1, NULL)
```

	Id	Ime	Prezime	Email	Pol	Godiste
1	1	Marko	Tosic	marko@db.com	1	1991
2	2	Milica	Jovicic	milica@db.com	2	1991
3	3	Theon	Greyjoy	theon@db.com	3	1991
4	4	Jovana	Peric	jovana@db.com	1	NULL

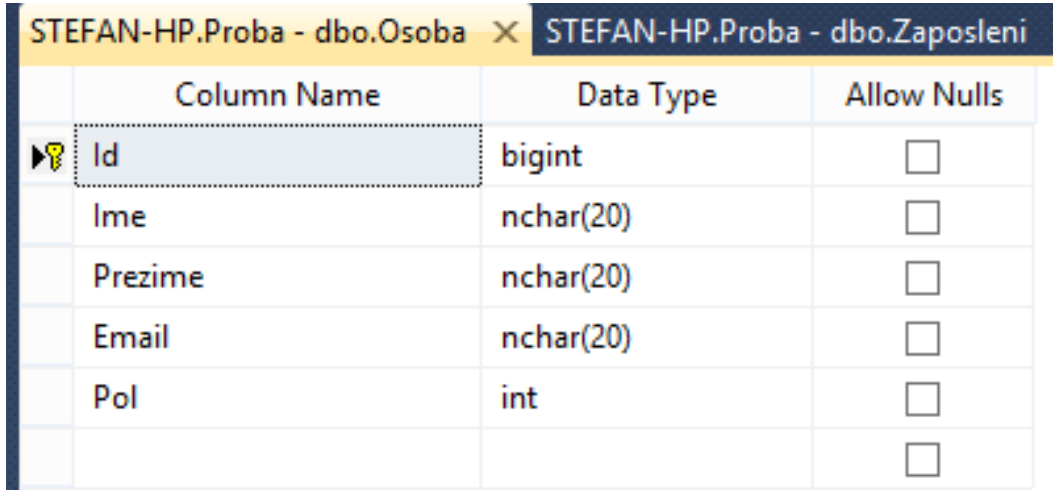
Rezultat upoređivanja NULL i neke poznaze vrednosti je Unknown što takođe prolazi Check constraint.

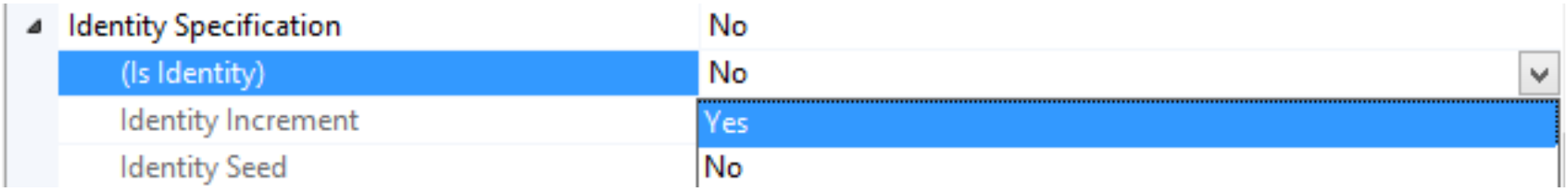
Iz tog razloga je dodat novi red u tabelu Osoba.

Identity column

Identity column

Ukoliko želimo da određena polja dobijaju automatski svoje vrednosti tada koristimo Identity column.





Identity Seed je inicjalna vrednost posmatrane kolone.

Identity Increment je vrednost koja se dodaje na prethodnu vrednost kako bi se dobila nova.

Identity column

Sada moržemo unositi nove redove bez specificiranja Id-a Osobe, jer će se on automatski postavljati (kolona Id je Identity column).

```
Insert into Osoba values ('Goran', 'Mitrovic', 'goran@db.com',1)
```

Results | Messages

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1

Identity column

Ukoliko želimo da postavimo eksplicitnu vrednost za kolonu koja je Identity column treba uraditi sledeće:

```
Set identity_insert Osoba on

Insert into Osoba(Id, Ime, Prezime, Email, Pol)
values (30,'Aleksandar', 'Kostic', 'aleksandar@db.com',1)

Insert into Osoba(Id, Ime, Prezime, Email, Pol)
values (25,'Jovan', 'Kostic', 'jovan@db.com',1)
```

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1
5	25	Jovan	Kostic	jovan@db.com	1
6	30	Aleksandar	Kostic	aleksandar@db.com	1

Identity column

Vraćanje na automatsko biranje vrednosti vrši se sa:

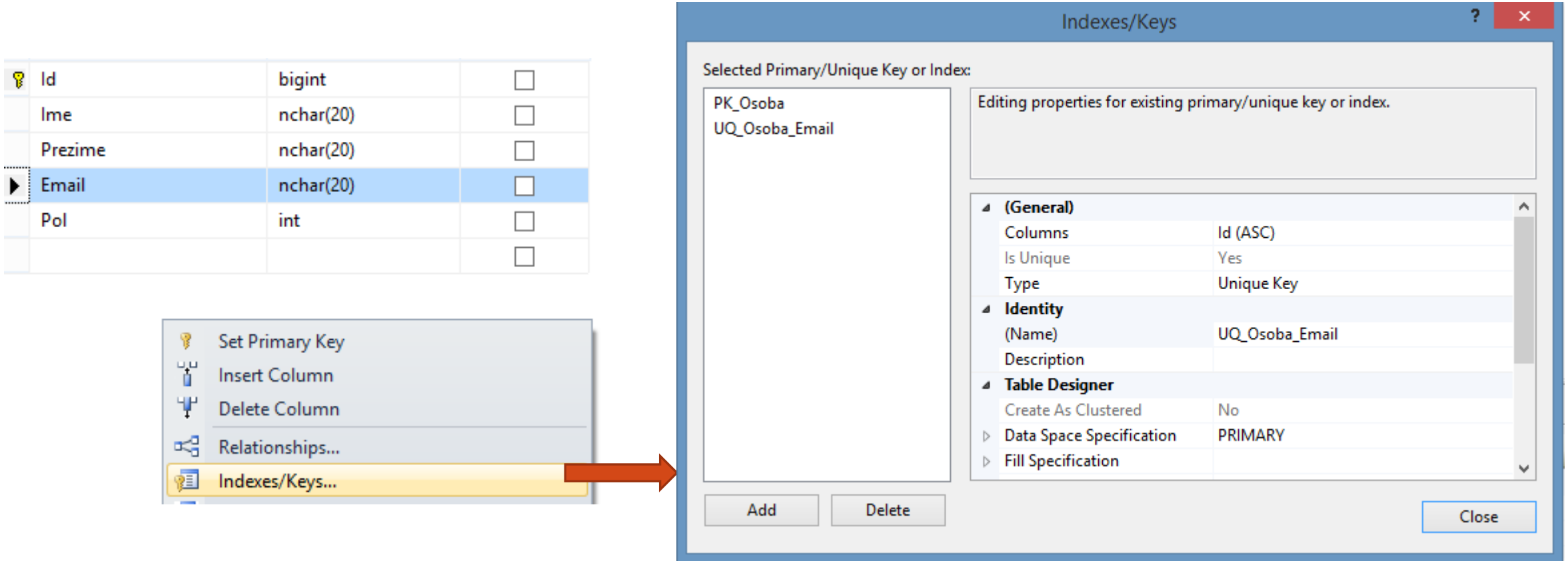
```
Set identity_insert Osoba off

Insert into Osoba values ('Jelena', 'Jovicic', 'jelena@db.com',2)
```

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1
5	25	Jovan	Kostic	jovan@db.com	1
6	30	Aleksandar	Kostic	aleksandar@db.com	1
7	31	Jelena	Jovicic	jelena@db.com	2

Za izmenu seed-a koristi se:

```
DBCC Checkident(Osoba, RESEED, 50)
```

Unique key constraint



Unique key constraint

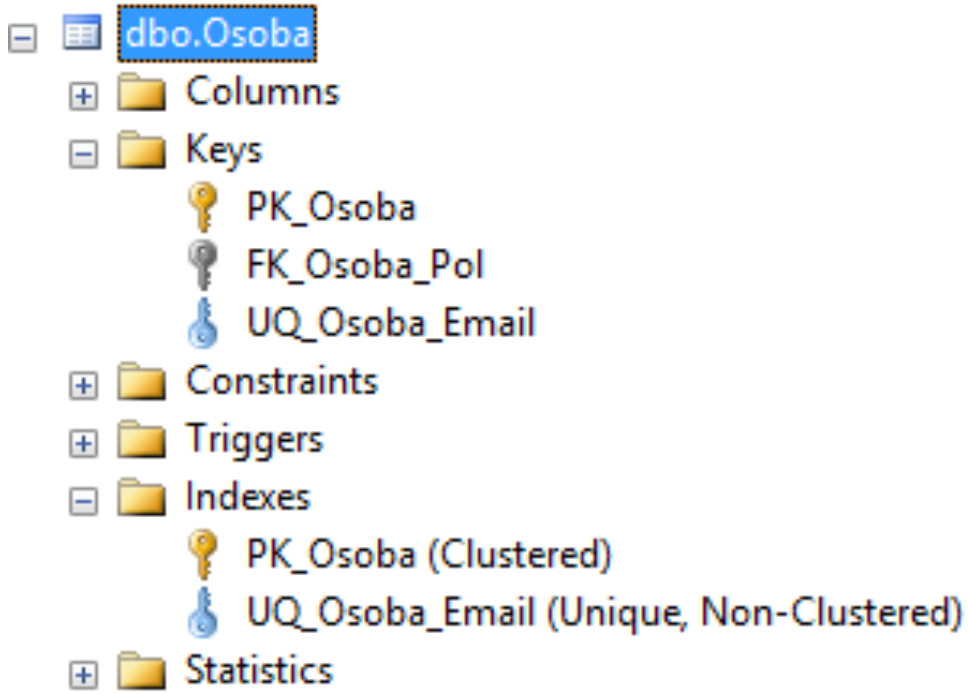
Isti efekat postižemo sledećom SQL skriptom:

```
Alter table Osoba
Add constraint UQ_Osoba_Email
Unique(Email)
```

U pozadini se stvara i Unique, Non-Clustered index.

Brisanje ograničenja:

```
Alter table Osoba
Drop Constraint UQ_Osoba_Email
```



Pogledi

Kreiranje pogleda:

```
Create view ZaposleniMuskarci
as
Select Id, Ime from Zaposleni
Where Pol = 'M'
```

Definiciju pogleda možemo videti koristeći:

```
Execute sp_helptext ZaposleniMuskarci
```

	Text
1	Create view ZaposleniMuskarci
2	as
3	Select Id, Ime from Zaposleni
4	Where Pol = 'M'

	Id	Ime	Plata	Pol
1	1	Ivan	3000	M
2	2	Milica	2000	Z
3	3	Marija	1000	Z

Tabela Zaposleni

	Id	Ime
1	1	Ivan

Pogled ZaposleniMuskarci

Pogledi

```
Create view OsobaPol
as
Select O.Ime, O.Prezime, O.Email, P.Tip
From Osoba O, Pol P
Where O.Pol = P.Id
```

	Id	Tip
1	1	Musko
2	2	Zensko
3	3	Nepoznato

Tabela Pol

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1

Tabela Osoba

	Ime	Prezime	Email	Tip
1	Marko	Tosic	marko@db.com	Musko
2	Milica	Jovicic	milica@db.com	Zensko
3	Theon	Greyjoy	theon@db.com	Nepoznato
4	Goran	Mitrovic	goran@db.com	Musko

Pogled OsobaPol

Pogledi

Pogledi se koriste:

- Radi skrivanja detaljnih informacija, i predstavljanja agregiranih podataka krajnjem korisniku
- Pojednostavljenja pisanja upita
- Kao mehanizam za implementaciju sigurnosti po redovima i kolonama (skrivanje redova i kolona)

Pogledi se ne čuvaju na disku kao tabele, već predstavljaju SQL kod koji se uvek izvršava kada izvršimo upit nad pogledom. (Paziti na optimizaciju!)

Kao i na tabele sledeći upiti se koriste i za poglede:

```
Alter View Zaposleni Statement
```

```
Drop View ViewName
```

Stored procedures

Umesto da pozivamo jedan isti upit više puta, možemo napraviti proceduru u koju ćemo smestiti TSQL (Transact SQL iskaze).

Kreiranje:

```
Create procedure ImeProcedure
Create proc ImeProcedure
```

Izvršavanje:

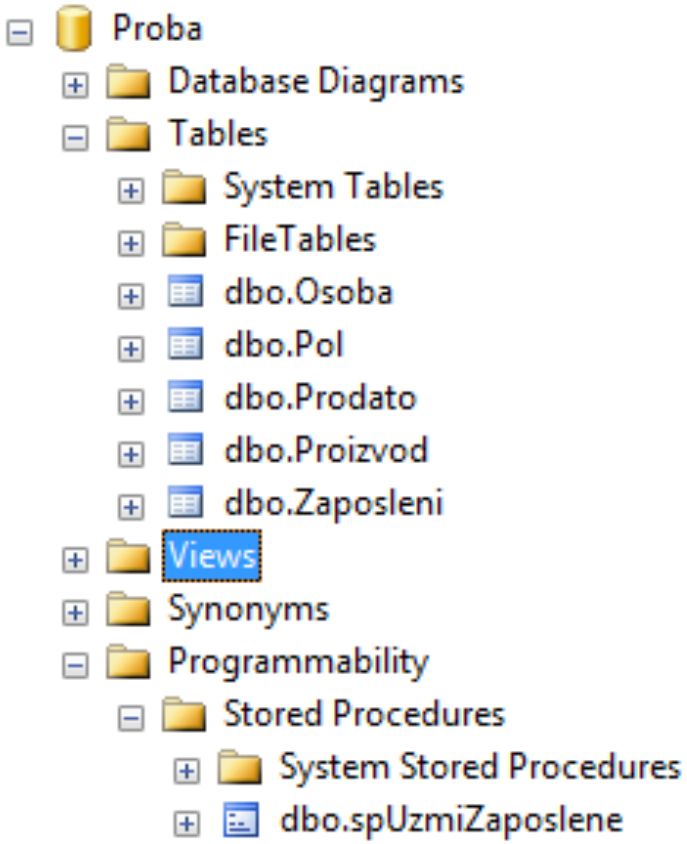
```
Execute ImeProcedure
Exec ImeProcedure
ImeProcedure
```

Stored procedures

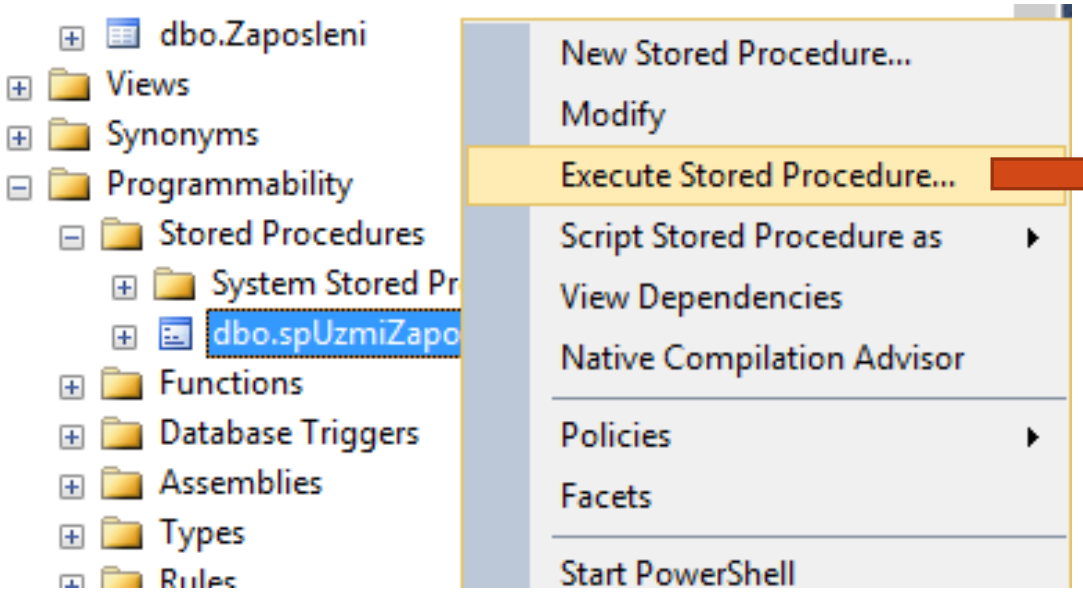
```
Create procedure spUzmiZaposlene
as
Begin
Select Ime, Pol
From Zaposleni
End
```

```
Execute spUzmiZaposlene
```

	Ime	Pol
1	Ivan	M
2	Milica	Z
3	Marija	Z



Stored procedures




```
USE [Proba]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[spUzmiZaposlene]

SELECT  'Return Value' = @return_value

GO
```

	Ime	Pol
1	Ivan	M
2	Milica	Z
3	Marija	Z

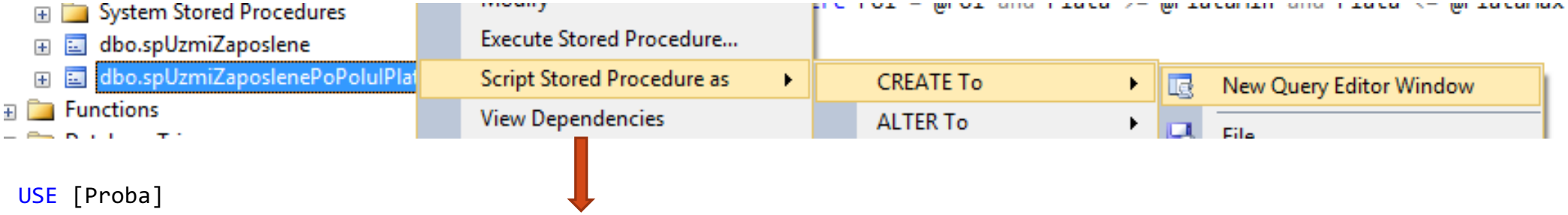
	Return Value
1	0

Stored procedures

Parametrozovane procedure

```
Create proc spUzmiZaposlenePoPoluIPlati
@Pol varchar(1),
@PlataMin int,
@PlataMax int
as
Begin
Select Ime, Pol, Plata
From Zaposleni
Where Pol = @Pol and Plata >= @PlataMin and
Plata <= @PlataMax
End


Exec spUzmiZaposlenePoPoluIPlati 'Z', 1500, 3000
```

Stored procedures




```
USE [Proba]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
Create proc [dbo].[spUzmiZaposlenePoPoluIPlati]
@Pol varchar(1),
@PlataMin int,
@PlataMax int
as
Begin
Select Ime, Pol, Plata
From Zaposleni
Where Pol = @Pol and Plata >= @PlataMin and Plata <= @PlataMax
End
GO
```

Stored procedures

Izmena procedure

```
Alter proc spUzmiZaposlenePoPoluIPlati
@Pol varchar(1),
@PlataMin int,
@PlataMax int
with encryption
as
Begin
Select Ime, Pol, Plata
From Zaposleni
Where Pol = @Pol and Plata >= @PlataMin and
Plata <= @PlataMax
Order by Plata
End
```

Brisanje procedure

```
Drop proc spUzmiZaposlene
```

Enkripcija onemogućavanje korisnika da vidi implementaciju.

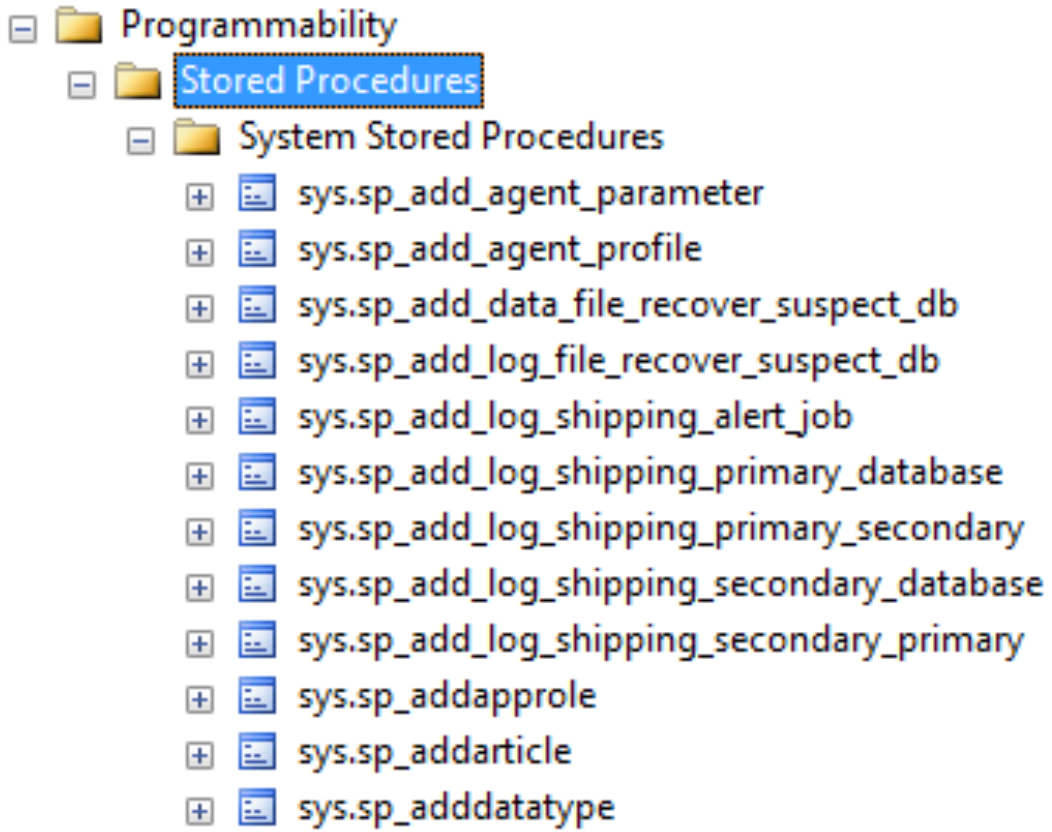
```
sp_helptext spUzmiZaposlenePoPoluIPlati

The text for object
'spUzmiZaposlenePoPoluIPlati' is
encrypted.
```

Stored procedures

Sistemske prcedure:

sp_renamedb
sp_helptext

Sve sistemske procedure počinju prefiksom **sp_** .



Stored procedures

Procedure mogu imati i **output** parametre i tako vraćati vrednost:

```
Create proc spBrojZaposlenihPoPolu
@Pol varchar(1),
@Broj int output
as
Begin
Select @Broj = Count(*)
From Zaposleni
Where Pol = @Pol
End
```

	Id	Ime	Plata	Pol
1	1	Ivan	3000	M
2	2	Milica	2000	Z
3	3	Marija	1000	Z

```
Declare @Br int
Execute spBrojZaposlenihPoPolu 'Z', @Br output
Print @Br
```

Deklariše promenljivu tipa int

Ispisiuje deklarisanu promenljivu kao poruku

Potrebno je navesti da je promenljiva izlazna inače će dobiti NULL vrednsot

```
Declare @Br int
Execute spBrojZaposlenihPoPolu @Pol = 'Z', @Broj = @Br output
SELECT @Br as N'@Broj'
```

	Broj
1	2

Ispisiuje deklarisanu promenljivu kao tabelu

Stored procedures

```
sp_help spBrojZaposlenihPoPolu
```

Daje informacije o tabelama, pogledima, procedurama...

	Name	Owner	Type	Created_datetime
1	spBrojZaposlenihPoPolu	dbo	stored procedure	2015-04-07 22:10:57.133

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1	@Pol	varchar	1	1	NULL	1	SQL_Latin1_General_CP1_CI_AS
2	@Broj	int	4	10	0	2	NULL

Lista sve zavisnosti objekta od drugih objekata (u ovom slučaju naša procedura zavisi od kolone Pol tabele Zaposleni)

```
sp_depends spBrojZaposlenihPoPolu
```

	name	type	updated	selected	column
1	dbo.Zaposleni	user table	no	yes	Pol

Stored procedures

Povratne vrednosti

```
Create proc spBrojZaposlenih
@Broj int output
as
Begin
Select Broj = Count(*)
From Zaposleni
End

Declare @Br int
Execute spBrojZaposlenih @Br output
Print @Br
```

```
Create proc spBrojZaposlenih
as
Begin
return
(Select Count(*)
From Zaposleni)
End

Declare @Br int
Execute @Br = spBrojZaposlenih
Print @Br
```

Povratne vrednosti se mogu koristiti samo kada vraćamo jednu promenljivu i to tipa int, dok se za ostale slučajeve moraju koristiti output parametri.

Stored procedures

Prednosti:

- Reupotrebljivost execution plana (za razliku od iskaza gde mala promena dovodi do stvaranja novog execution plana, procedure mogu uvek koristiti execution plan koji je prvi put kreiran)
- Olakšano održavanje koda (izmene samo na jednom mestu)
- Smanjen mrežni saobraćaj (za razliku od običnih upita koji se celi moraju slati preko mreže, ovde se šalju samo naziv procedure, parametri i komanda za izvršavanje)
- Bolja sigurnost (slično pogledima, korisniku se može dodeliti pristup procedurama, samim tabelama)
- Izbegavanje SQL-Injection napada

Functions

Funkcije mogu biti:

- Scalar functions
- Inline-table valued functions
- Multi-statement table-valued functions

Scalar functions su funkcije koje mogu, ali ne moraju imati parametre i koje uvek vraćaju jednu skalarnu vrednost. Ta povratna vrednost može biti bilo šta osim **text, ntext, image, cursor, timestamp**.

```
Create function fName(@Parametar1 tip1, @Parametar2 tip2, ...)
returns returnValType
as
Begin
Statements
return returnVal
End
```

Functions

Inline-table valued functions su funkcije čiji je povratni tip tabela. Nemaju Begin i End već se povratna vrednost mora odrediti jednim upitom.

```
Create function fName(@Parametar1 tip1, @Parametar2 tip2, ...)
returns TABLE
as
return StatementThatReturnTable
```

```
Use Proba
Create function fZaposleniPoPolu(@Pol
varchar(1))
returns TABLE
as
return (Select *
From Zaposleni
Where Pol = @Pol)
```

```
Select * From dbo.fZaposleniPoPolu('Z') Where Plata > 1000
```

	Id	Ime	Plata	Pol
1	2	Milica	2000	Z

Functions

Multi statement table valued functions su funkcije čiji je povratni tip tabela čije kolone možemo definisati u funkciji. Zbog postojanja više upita potrebno je koristiti Begin i End. Tip povratne vrednosti je TABLE.

```
Create function fName(@Pol varchar(1))
returns @Table TABLE(Ime varchar(20), Plata
int, Pol varchar(1))
as
Begin
Insert into @Table
Select Ime, Plata, Pol
From Zaposleni
Where Pol = @Pol

return
End
```

```
Select * from dbo.fName('Z') Where Plata > 1000
```

	Id	Ime	Plata	Pol
1	2	Milica	2000	Z

Functions

Moguće je UPDATE-ovati tabele od kojih zavisi Inline-table valued funkcija, dok to nije moguće za Multi statement table valued funkcije.

```
Select * from Zaposleni
```

	Id	Ime	Plata	Pol
1	1	Ivan	3000	M
2	2	Milica	2000	Z
3	3	Marija	1000	Z

```
Select * from Zaposleni
```

	Id	Ime	Plata	Pol
1	1	Ivan	3000	M
2	2	Jelena	2000	Z
3	3	Marija	1000	Z

```
Update fZaposleniPoPolu('Z') Set Ime = 'Jelena' Where Id = 2
```

I kod funkcija se, ukoliko je potrebno, može koristiti „With Encryption“.

Privremene tabele

Obične tabele:

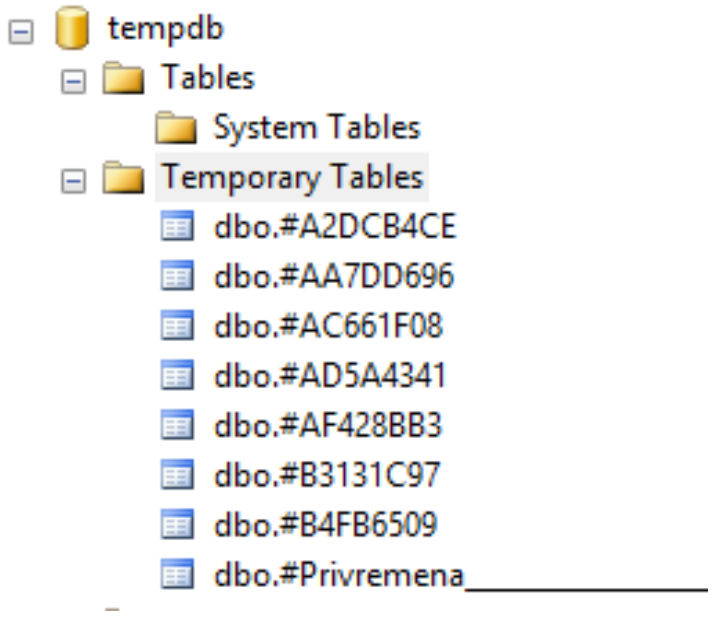
- Kreirane sa CREATE TABLE NazivTabele
- Brisanje sa DROP TABLE NazivTabele
- Nalaze se u folderu **Tables** u bazi u kojoj je kreirana

Privremene tabele:

- Kreirane sa CREATE TABLE #NazivTabele
- Može eksplicitno iskazom DROP TABLE, ali karakteristično za nju je da ona postoji sve dok postoji konekcija koja ju je napravila (ukoliko zatvorimo .sql fajl u Management Studiu privremena tabela će nestati jer se zatvaranjem ovog fajla zatvorila i konekcija)
- Raspoloživa je samo za konekciju koja ju je kreirala
- Nalaze se u folderu **Temporary Tables** sistemske baze **tempdb**

Privremene tabele

```
Create table #Privremena
(
Id int Primary Key,
Ime varchar(20)
)
Insert Into #Privremena values (1,'Prvi')
Insert Into #Privremena values (2,'Drugi')
```



Proveravanje da li je privremena tabela napravljena upitom:

```
Select name from tempdb..sysobjects
Where name like '#Privremena%'
```

```
Select * from #Privremena
```

Prva konekcija:

	Id	Ime
1	1	Prvi
2	2	Drugi

Druga konekcija:

```
Msg 208, Level 16, State 0, Line 1
Invalid object name '#Privremena'.
```

Privremene tabele

Ukoliko se privremena tabela napravi unutar procedure njena instanca nestaje kada se izvršavanje procedure završi.

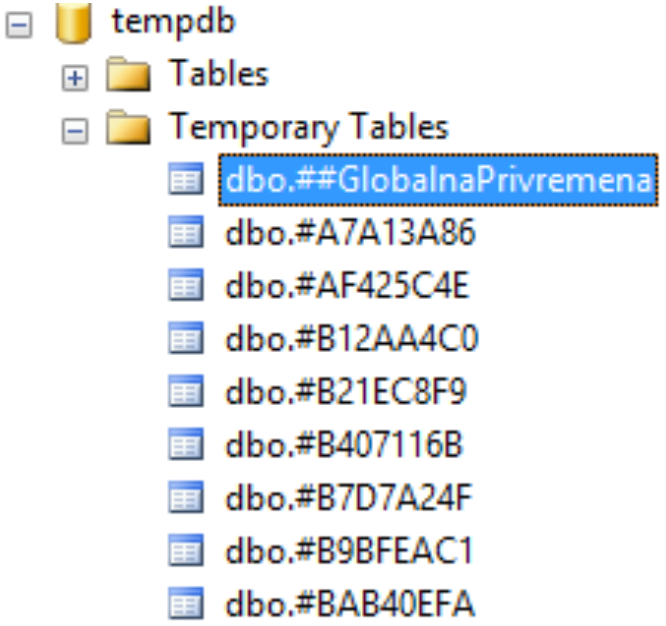
```
Create proc P
As
Begin
Create table #Privremena
(
Id int Primary Key,
Ime varchar(20)
)
Insert Into #Privremena values (1,'Prvi')
Insert Into #Privremena values (2,'Drugi')
Select * from #Privremena
End

execute P
```

Kada napravimo istu privremenu tabeli iz različitih konekcija napraviće se tabela sa istim početnim delom naziva, a različitim automatski generisanim brojem na kraju.

Privremene tabele

Globalne privremene tabele se kreiraju koristieći prefiks ##.

```
Create table ##GlobalnaPrivremena
(
Id int Primary Key,
Ime varchar(20)
)
```



Ovakve tabele nemaju izgenerisan broj na kraju jer njeno ime mora biti jedinstveno. Moguće ih je koristititi iz svih konekcija, i one se brišu kada nestane poslednja konekcija sa bazom.

Triggers

Trigeri mogu biti:

- DML trigeri
- DDL trigeri
- Logon trigeri

DML trigeri se dele na:

- After (For)
- Instead of

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::= [ ENCRYPTION ] [ EXECUTE AS Clause ]
<method_specifier> ::=  assembly_name.class_name.method_name
```

After (For) okidači se okidaju pošto se dogodi INSERT, UPDATE ili DELETE u zavisnosti od toga šta je specificirano u definiciji okidača.

Instead of okidači se okidaju neposredno pre nego što se dogodi INSERT, UPDATE ili DELETE i oni zamenjuju njihove akcije (INSERT, UPDATE ili DELETE se ne izvrše već se izvrši telo okidača).

Cursor

```
DECLARE @MyCursor CURSOR;
DECLARE @MyField YourFieldDataType;
BEGIN
   SET @MyCursor = CURSOR FOR
   select top 1000 YourField from dbo.table
      where StatusID = 7

   OPEN @MyCursor
   FETCH NEXT FROM @MyCursor
   INTO @MyField

   WHILE @@FETCH_STATUS = 0
   BEGIN
   // YOUR ALGORITHM GOES HERE
    FETCH NEXT FROM @MyCursor
    INTO @MyField
   END;

   CLOSE @MyCursor ;
   DEALLOCATE @MyCursor;
END;
```

Koriste se za iterativni prolazak kroz redove tabele i daju dobre performanse.

@@FETCH_STATUS

Vraća status poslednjeg kursor FETCH iskaza izdatim nad bilo kom kursoru otvorenim trenutnom konekcijom.

Rad sa datumima

Primeri funkcija koje se mogu koristiti sa podacima koji su tipa datumi:

```
select MONTH(getdate())

select YEAR(CURRENT_TIMESTAMP)

SELECT DATEADD(month, 1, '01/10/2023')

SELECT CONVERT(getdate())

select DATEDIFF(year, '01/10/2023',getdate())
```

Rad sa datumima

```
CREATE FUNCTION izracunajBrojGodinaV1
(
    @DatumRodjenja Date
)
RETURNS int
AS
BEGIN
  declare @Godina int
  set @Godina =
    DATEDIFF(year, @DatumRodjenja,getdate())
  if(MONTH(@DatumRodjenja)>MONTH(getdate()) OR
    (MONTH(@DatumRodjenja)=MONTH(getdate())
    AND DAY(@DatumRodjenja)>DAY(getdate())))
  set @Godina = @Godina - 1
  return @Godina
END
```

```
CREATE FUNCTION izracunajBrojGodinaV2
(
    @DatumRodjenja Date
)
RETURNS int
AS
BEGIN
  return DATEDIFF(year, @DatumRodjenja,getdate())
    - CASE WHEN
      MONTH(@DatumRodjenja)>MONTH(getdate()) OR
      (MONTH(@DatumRodjenja)=MONTH(getdate()) AND
      DAY(@DatumRodjenja)>DAY(getdate()))
      THEN 1
      ELSE 0
    END
END
```

Zadatak 1

Data je baza podataka Test sa sledećim relacijama:
student_mast(STUDENT_ID, NAME, ST_CLASS)
student_marks(STUDENT_ID, NAME,SUB1, SUB2, SUB3, SUB4, SUB5, TOTAL, PER_MARKS, GRADE)
student_log(USER_ID, DESCRIPTION)

Pretpostavimo da tabela student_marks sadrži podatke o studentima koji su se prijavili za polaganje testa. Kako test još nije završen, pretpostavimo da se u tabeli za svakog studenta nalaze samo vrednosti pripadajućih atributa STUDENT_ID i NAME dok su vrednosti svih ostalih atributa postavljene na podrazumevane vrednosti (0 ili NULL). Završen je test i studenti su dobili ocene za 5 različitih predmeta. Potrebno je izmeniti podatke o studentima u tabeli student_marks, pri čemu je za svakog studenta potrebno uneti ocene koje je dobio na testu za svaki od 5 predmeta. U skladu sa dobijenim ocenama potrebno je automatski izmeniti ukupnu ocenu za sve predmete (TOTAL), prosečnu ocenu (PER_MARKS) i ukupan uspeh na testu (GRADE). Vrednosti ovih atributa se izračunavaju na sledeći način:
Ukupna ocena: TOTAL = SUB1 + SUB2 + SUB3 + SUB4 + SUB5
Prosečna ocena: PER_MARKS= (TOTAL)/5
Uspeh: GRADE= - If PER_MARKS>=90 -> ‘Odlican’ - If PER_MARKS>=75 AND PER_MARKS ‘Vrlo dobar’ - If PER_MARKS>=60 AND PER_MARKS ‘Dobar’ - If PER_MARKS>=40 AND PER_MARKS ‘Dovoljan’ - If PER_MARKS ‘Nedovoljan’

Zadatak 1

```
Insert into student_marks(STUDENT_ID,NAME)
values (1,'a')

Update student_marks set SUB1 = 1, SUB2 = 2,
SUB3 = 3, SUB4 = 4, SUB5 = 5
where STUDENT_ID = 1



CREATE TRIGGER Ocene
     ON student_marks
     FOR INSERT, UPDATE
     AS
BEGIN
     Declare @stud_id int
     Declare @prosek float
     Declare @uspeh nchar(10)
     Declare @MyCursor Cursor

     SET @MyCursor = CURSOR FOR
     select STUDENT_ID,
          (SUB1 + SUB2 + SUB3 + SUB4 + SUB5)/5
     from inserted

     FETCH NEXT FROM @MyCursor
     INTO @stud_id, @prosek
     WHILE @@FETCH_STATUS = 0
     BEGIN
          set @uspeh = case
               when @prosek>=9 then 'Odlican'
               when @prosek>=8 then 'Vrlo dobar'
               when @prosek>=7 then 'Dobar'
               when @prosek>=6 then 'Dovoljan'
               else 'Nedovoljan'
          end
          print(@suma)
          print(@prosek)
          print(@uspeh)

          update student_marks
          set TOTAL = @suma,
              PER_MARKS = @prosek, GRADE = @uspeh
          where STUDENT_ID = @stud_id

          FETCH NEXT FROM @MyCursor
          INTO @stud_id, @suma
     END

     CLOSE @MyCursor
     DEALLOCATE @MyCursor
End
```

Zadatak 2

Data je baza Test iz prethodnog zadatka. Potrebno je za sve studente iz tabele student_mast promeniti razred u sledeći. Kreirati triger koji proverava da uneti razred nije manji od starog razreda. Kreirati triger koji će obezbediti da se posle svake izmene pojedinačne n-torke u tabeli student_mast doda nova n-torka u tabelu student_log sa informacijom o korisniku (user_id) koji je izvršio izmenu i opisom u vezi izvršene izmene. Kreirati i triger koji nakon svakog brisanja n-torke iz tabele student_mast dodaje novu vrstu u tabelu student_log sa informacijom o korisniku koji vrši brisanje i kratkim opisom izvršenog brisanja.

Zadatak 2

```
Update student_mast Set ST_CLASS = 2
Where STUDENT_ID = 1

CREATE TRIGGER NoviRazred
     ON student_mast
     FOR UPDATE
     AS
BEGIN
      Declare @poruka varchar(40)
      Declare @id int
      Declare @stariRazred int
      Declare @noviRazred int
      Declare @MyCursorI Cursor
      Declare @MyCursorD Cursor

      SET @MyCursorD = CURSOR FOR
      select ST_CLASS, STUDENT_ID
      from deleted
      SET @MyCursorI = CURSOR FOR
      select ST_CLASS
      from inserted

      OPEN @MyCursorI
      OPEN @MyCursorD
      FETCH NEXT FROM @MyCursorD
      INTO @stariRazred, @id

      FETCH NEXT FROM @MyCursorI
      INTO @noviRazred

      WHILE @@FETCH_STATUS = 0
      BEGIN
          if (@noviRazred < @stariRazred)
          BEGIN
              print('Novi razred je manji od starog.’)
              rollback transaction
          END
          else
          begin
              set @poruka = CONCAT('Student ciji je id: ',
                  @id, ' je presao iz razreda ', @stariRazred,
                  ' u razred ‘, @noviRazred, '.’)
              insert into student_log values (1, @poruka)
          end
          FETCH NEXT FROM @MyCursorD
          INTO @stariRazred, @id
          FETCH NEXT FROM @MyCursorI
          INTO @noviRazred
      END
      CLOSE @MyCursorI
      DEALLOCATE @MyCursorI
      CLOSE @MyCursorD
      DEALLOCATE @MyCursorD
End
```

Zadatak 2

```
CREATE TRIGGER BrisanjeStudenta
    ON student_mast
    FOR DELETE
    AS
BEGIN
    insert into student_log
    select STUDENT_ID,
       CONCAT('Student ciji je id: ', STUDENT_ID, ' je izbrisan'))
    from deleted
END
```

Zadatak 3

Data je relacija emp_details kao deo baze ljudskih resursa:
emp_details(EMPLOYEE_ID, NAME, SALARY , COMMISSION_PCT)

Kreirati triger koji omogućava da se nakon svakog unošenja podataka o zaposlenima u tabelu emp_details izvrši provera njihove zarade i ako je zarada veća od 20000 da se za tog zaposlenog postavi provizija od urađenog posla (COMMISSION_PCT) na 0.1 a u suprotnom na 0.5.

Zadatak 3

```
CREATE TRIGGER trigger_plata
    ON emp_details
    FOR Insert
    As
Begin
    UPDATE emp_details
    SET COMMISSION_PCT = 0.1
    WHERE EMPLOYEE_ID in
        (SELECT EMPLOYEE_ID FROM inserted where SALARY>=2000)

    UPDATE emp_details
    SET COMMISSION_PCT = 0.5
    WHERE EMPLOYEE_ID in
        (SELECT EMPLOYEE_ID FROM inserted where SALARY<2000)
End
```

```
Insert into emp_details(NAME, SALARY) values
('Sasa', 50000)
```

Zadatak 4

Data je aproksimacija izdavačke relacione baze sa sledećom šemom:
P (P_SIF, IME, BR_NASLOVA, DRZAVA)
I (I_SIF, NAZIV, STATUS, DRZAVA)
K (K_SIF, NASLOV, OBLAST)
KP (K_SIF, P_SIF, R_BROJ)
KI (K_SIF, I_SIF, IZDANJE, GODINA, TIRAZ)

Prve tri relacije predstavljaju tipove entiteta PISAC, IZDAVAC i KNJIGA redom. Relacija KP predstavlja apstraktni tip entiteta AUTOR, tj. odnos između tipova entiteta KNJIGA i PISAC, dok relacija KI predstavlja apstraktni tip entiteta IZDAVASTVO, tj. odnos između tipova entiteta KNJIGA i IZDAVAC.

Prilikom registracije novog autorskog dela, odnosno prilikom unošenja n-torke u relaciju KP, uvećati vrednost atributa BR_NASLOVA za jedan u okviru relacije P za pisca koji je autor unetog autorskog dela. Pobrinuti se da se BR_NASLOVA smanji kada se briše n-torka iz relacije KP.

Zadatak 4

```
Insert into K values ('naslov1', 'oblast1')
Insert into P values ('ime1', 0, 'oblast1')
Insert into KP values (1, 1, 1)
```

```
CREATE TRIGGER trigger_brNaslova
      ON KP
      FOR Insert, Update, Delete
      As
Begin
      Declare @pSif int
      Declare @MyCursor Cursor

      SET @MyCursor = CURSOR FOR
      Select P_SIF From inserted
      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor INTO @pSif

      WHILE @@FETCH_STATUS = 0
      BEGIN
          UPDATE P
          SET BR_NASLOVA=BR_NASLOVA+1
          WHERE P_SIF = @pSif
          FETCH NEXT FROM @MyCursor INTO @pSif
      END
      CLOSE @MyCursor
      DEALLOCATE @MyCursor

      SET @MyCursor = CURSOR FOR
      Select P_SIF From deleted
      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor INTO @pSif

      WHILE @@FETCH_STATUS = 0
      BEGIN
          UPDATE P
          SET BR_NASLOVA=BR_NASLOVA+1
          WHERE P_SIF = @pSif
          FETCH NEXT FROM @MyCursor INTO @pSif
      END
      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```

Zadatak 5

Kreirati funkciju „izracunajStarost“ koja kao parametar prima godinu rođenja i vraća starost osobe u godinama.

```
create function izracunajStarost(@DatumRodj Date)
    returns int
as
Begin
  Declare @Godine int
  Set @Godine = DATEDIFF(YEAR, @DatumRodj, GETDATE()) - Case
      When (MONTH(@DatumRodj) > MONTH(GETDATE())) or
      (MONTH(@DatumRodj)=MONTH(GETDATE()) and DAY(@DatumRodj)>DAY(GETDATE()))
      Then 1
      Else 0
    End
  Return @Godine
End


Select dbo.izracunajStarost('08/25/1991') As Godine
```

Zadatak 6

Prosek roka se određuje po završetku ispitnog roka, kada je njegov status 'F'-finaliziran. Okidač ProsekRoka treba da reaguje na UPDATE tabele ROK, i to kolone Status u novu vrednost 'F'. Ovo razrešava i situacije naknadnih ispravki ocena, pošto se u toj situaciji kolona Status postavlja na vrednost 'Z' a nakon ispravki ponovo na vrednost 'F'. U računanje proseka roka ulaze i prolazne ocene koje su naknadno poništene.

Zadatak 6

```
Create TRIGGER trigger_ProsekRoka
        ON ROK
        FOR Update
        As
Begin
        Declare @IdRok int
        Declare @MyCursor Cursor

        SET @MyCursor = CURSOR FOR
        Select IDRok
        From inserted I, deleted D
        where I.IDRok=D.IDRok AND I.Status = 'F' AND D.Status <> 'F'

        OPEN @MyCursor
        FETCH NEXT FROM @MyCursor INTO @IdRok

        WHILE @@FETCH_STATUS = 0
        BEGIN
                --Execute spProsekRoka @IdRok
                --Execute spProsekRokaWithOneSelect @IdRok
                Execute spProsekRokaUsingCursor @IdRok
                FETCH NEXT FROM @MyCursor INTO @IdRok
        END

        CLOSE @MyCursor
        DEALLOCATE @MyCursor
End
```

Zadatak 6

```
Create proc spProsekRoka
      @IdRok int
      as
Begin
      Declare @IDIsp
      Declare @ZbirOcena int
      Declare @BrojOcena int
      Declare @Broj int
      Declare @Zbir int
      Declare @Prosek int

      SET @BrojOcena = 0
      SET @ZbirOcena = 0

      SELECT @IDIsp = MIN(IDIsp)
      FROM ISPIT
      WHERE IDRok = @IdRok

      While(@IDIsp IS NOT NULL)
      Begin
            SELECT @Broj =COUNT(*), @Zbir =SUM(Ocena)
            FROM POLAGANJE
            WHERE IDIsp = @IDIsp AND Ocena > 5

            SET @BrojOcena = @BrojOcena
                  + coalesce(@Broj,0)
            SET @ZbirOcena = @ZbirOcena
                  + coalesce(@Zbir,0)

            select @IDIspN = MIN(IDIsp)
            FROM ISPIT
            WHERE IDRok = @IdRok AND IDIsp > @IDIsp
      End

      IF (@BrojOcena > 0)
            @Prosek = @ZbirOcena/@BrojOcena
      ELSE
            @Prosek = 0

      UPDATE ROK
      SET ProsekOcena = @Prosek
      WHERE IDRok = @IdRok
End
```

Zadatak 6

```
Create proc spProsekRokaUsingCursor
      @IdRok int
      as
Begin
      Declare @IDIsp
      Declare @ZbirOcena int
      Declare @BrojOcena int
      Declare @Broj int
      Declare @Zbir int
      Declare @Prosek int
      Declare @MyCursor CURSOR

      SET @BrojOcena = 0
      SET @ZbirOcena = 0

      SET @MyCursor = CURSOR FOR
      SELECT IDIsp
      FROM ISPIT
      WHERE IDRok = @IdRok

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor
      INTO @IDIsp

      While @@FETCH_STATUS = 0
      Begin
            SELECT @Broj =COUNT(*), @Zbir =SUM(Ocena)
            FROM POLAGANJE
            WHERE IDIsp = @IDIsp AND Ocena > 5

            SET @BrojOcena = @BrojOcena
                  + coalesce(@Broj,0)
            SET @ZbirOcena = @ZbirOcena
                  + coalesce(@Zbir,0)

            FETCH NEXT FROM @MyCursor
            INTO @IDIsp
      End
      CLOSE @MyCursor
      DEALLOCATE @MyCursor

      IF (@BrojOcena > 0)
            @Prosek = @ZbirOcena/@BrojOcena
      ELSE
            @Prosek = 0

      UPDATE ROK
      SET ProsekOcena = @Prosek
      WHERE IDRok = @IdRok
End
```

Zadatak 6

```
Create proc spProsekRokaWithOneSelect
      @IdRok int
      as
Begin
      Declare @IDIsp
      Declare @ZbirOcena int
      Declare @BrojOcena int
      Declare @Prosek int

      SELECT @BrojOcena = COUNT(*),
            @ZbirOcena =SUM(Ocena)
      FROM POLAGANJE P, ISPIT I
      WHERE I.IDRok = @IdRok AND P.IDIsp=I.IDIsp
            AND Ocena > 5

      IF (@BrojOcena IS NOT NULL)
            @Prosek = @ZbirOcena/@BrojOcena
      ELSE
            @Prosek = 0

      UPDATE ROK
      SET ProsekOcena = @Prosek
      WHERE IDRok = @IdRok
End
```

Zadatak 7

Posmatrani sistem je agencija za drumski prevoz tereta koja raspolaže sa vozačima i kamionima koji su svi iste nosivosti i koja osplužuje svoje komitente koji su ili firme ili osobe. Agencija vrši prevoz pošiljki na linijama od početnog do zadnjeg mesta, pri čemu te linije prolaze kroz usputna mesta. Komitenti ostavljaju svoje pošiljke u mestima utovara, a agencija ih prevozi odatle do mesta istovara vožnjama na kojima su angažovani vozači i kamioni. Mesto utovara može biti i početno, a mesto istovara i zadnje mesto na liniji. Pošiljka se prima samo ako je moguće ostvariti njen prevoz. Prilikom zadržavanja u usputnom mestu sa kamiona se prvo istovaruju pošiljke za to mesto, a zatim se vrši utovar pošiljki koje su predate tom mestu. Utovar se vrši po redosledu predaje.

Zadatak 7

Šema RBP:

KAMION (IDKam, Marka,Nosivost)
VOZAC (IDVoz, Kategorija)
KOMITENT (IDKom, Naziv, Adresa)
FIRMA (IDKom, ZiroRacun)
OSOBA (IDKom, JMBG)
MESTO (IDMes, Naziv)
LINIJA (IDLin, Duzina, IDMesOd, IDMesDo)
POSILJKA (IDPos, Tezina, DatumVremePredaje, Status, IDKom, IDMesUto, IDMesIst)
VOZNJA (IDVzn, DatumVremeOd, DatumVremeDo, Status, IDKam, IDVoz, IDLin)
PREVOZ (IDPre, IDVzn, IDPos)
VOZI (IDVoz, IDKam)
PREVOZI_SE (IDPos, IDVzn)
JE_U_MESTU (IDVzn, IDMes)
JE_USPUTNO (IDLin, IDMes, Redosled)

Zadatak 7

Napomene:

1. Vožnja se evidentira u trenutku početka, pri čemu atribut DatumVremeDo dobija vrednost NULL i atribut Status vrednost 'P' (početo).
2. U trenutku završetka vožnje atribut DatumVremeDo dobija odgovarajuću vrednost i atribut Status dobija vrednost 'Z' (završeno).
3. Između dva mesta može postojati više linija (alternativni putevi).
4. Status pošiljke po predaji postaje 'P', po utovaru 'U' a po istovaru 'I'.

Zadatak 7

Sastaviti jedan ili više SQL okidača koji u tabeli KAMION održavaju redudantne podatke o ukupnom broju vožnji i ukupnoj pređenoj kilometraži (UkupnoVoznji, UkupnoKm) u trenutku izmene podatka o statusu vožnje (kada status dobije vrednost Z).

Zadatak 7

```
Create TRIGGER DetaljiKamion
     ON Voznja
     FOR Update
     As
Begin
     Declare @MyCursor CURSOR
     Declare @IDKam int
     Declare @Status varchar(1)
     Declare @IDLin int
     Declare @Duzina int

     SET @MyCursor = CURSOR FOR
     Select Status, IDLin, IDKam
     From inserted
     Where Status= 'Z'

     OPEN @MyCursor
     FETCH NEXT FROM @MyCursor
     INTO @Status, @IDLin, @IDKam

     WHILE @@FETCH_STATUS = 0
     BEGIN
          Select @Duzina = Duzina
          From Linija
          Where IDLin = @IDLin

          Update Kamion
          Set UkupnoKm = UkupnoKm + @Duzina,
          Set UkupnoVoznji = UkupnoVoznji + 1
          Where IDKam = @IDKam

          FETCH NEXT FROM @MyCursor
          INTO @Status, @IDLin, @IDKam
     END

     CLOSE @MyCursor
     DEALLOCATE @MyCursor
End
```

Zadatak 7

Sastaviti jedan ili više SQL okidača koji u tabeli LINIJA održavaju redudantne podatke o nazivima mesta (NazivOd, NazivDo) i ukupnom broju vožnji (BrojVoznji) na posmatranoj liniji.

Zadatak 7

```
Create TRIGGER DetaljiLinija1
      ON Linija
      FOR Insert
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDLin int
      Declare @IDMesOd int
      Declare @IDMesDo int
      Declare @NazivOd varchar(20)
      Declare @NazivOd varchar(20)

      SET @MyCursor = CURSOR FOR
      Select IDMesOd, IDMesDo, IDLin
      From inserted

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor
      INTO @IDMesOd, @IDMesDo, @IDLin

      WHILE @@FETCH_STATUS = 0
      BEGIN
            Select @NazivOd = Naziv
            From Mesto
            Where IDMes = @IDMesOd

            Select @NazivDo = Naziv
            From Mesto
            Where IDMes = @IDMesDo

            Update Linija
            Set NazivOd = @NazivOd, NazivOd = @NazivOd,
                  BrojVoznji = 0
            Where IDLin = @IDLin ;

            FETCH NEXT FROM @MyCursor
            INTO @IDMesOd, @IDMesDo, @IDLin
      END

      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```

Zadatak 7

```
Create TRIGGER DetaljiLinija2
     ON Voznja
     FOR Update
     As
Begin
     Declare @MyCursor CURSOR
     Declare @IDLin int

     SET @MyCursor = CURSOR FOR
     Select IDLin
     From inserted
     Where Status= 'Z'

     OPEN @MyCursor
     FETCH NEXT FROM @MyCursor
     INTO @IDLin

     WHILE @@FETCH_STATUS = 0
     BEGIN
          Update Linija
          Set BrojVoznji = BrojVoznji + 1
          Where IDLin = @IDLin

          FETCH NEXT FROM @MyCursor
          INTO @IDLin
     END

     CLOSE @MyCursor
     DEALLOCATE @MyCursor
End
```

Zadatak 7

Sastaviti jedan ili više SQL okidača koji u tabeli MESTO održavaju redudantne podatke o broju i težini poslatih pošiljki (BrojPoslato, TezinaPoslato) iz posmatranog mesta.

Zadatak 7

```
Create TRIGGER DetaljiMesto
     ON Prevozi_se
     FOR Delete
     As
Begin
     Declare @MyCursor CURSOR
     Declare @IDPos int
     Declare @Tezina int
     Declare @IDMesOd int

     SET @MyCursor = CURSOR FOR
     Select IDPos
     From deleted

     OPEN @MyCursor
     FETCH NEXT FROM @MyCursor
     INTO @IDPos

     WHILE @@FETCH_STATUS = 0
     BEGIN
          Select @IDMesOd = IDMesOd,
               @Tezina = Tezina
          From Posiljka
          Where IDPos = @IDPos

          Update Mesto
          SET BrojPoslato = BrojPoslato + 1,
            TezinaPoslato = TezinaPoslato +
          @Tezina
          Where IDMes = @IDMesOd;

          FETCH NEXT FROM @MyCursor
          INTO @IDIsp
     END

     CLOSE @MyCursor
     DEALLOCATE @MyCursor
End
```

Zadatak 7

Sastaviti jedan ili više SQL okidača koji u tabeli VOZNJA održavaju redudantne podatke o nazivima mesta (NazivOd, NazivDo) i imenu vozača (Ime) u trenutku izmene podatka o status vožnje (kada status dobije vrednost Z).

Zadatak 7

```
Create TRIGGER DetaljiVoznja
      ON Voznja
      FOR Update
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDVzn int
      Declare @IDMesOd int
      Declare @IDMesDo int
      Declare @IDVoz int

      SET @MyCursor = CURSOR FOR
      Select IDVzn, IDLin
      From inserted
      Where Status= 'Z'

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor
      INTO @IDVzn, @IDLin, @IdVoz

      WHILE @@FETCH_STATUS = 0
      BEGIN
            SELECT @IDMesOd = IDMesOd,
                  @IDMesDo = IDMesDo
            FROM Linija
            WHERE IDLin = @IDLin

            UPDATE Voznja
            SET
              NazivOd = (SELECT Naziv
                  FROM Mesto WHERE IDMes = @IDMesOd),
              NazivDo = (SELECT Naziv
                  FROM Mesto WHERE IDMes = @IDMesDo),
              Ime = (SELECT Ime
                  FROM Vozac WHERE IDVoz = @IDVoz)
            WHERE IDVzn = @IDVzn

            FETCH NEXT FROM @MyCursor
            INTO @IDVzn, @IDLin, @IdVoz
      END

      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```

Zadatak 7

Sastaviti jedan ili više SQL okidača koji u tabeli VOZAC_KOMITENT održavaju redudantne podatke za potrebe pregleda 'ImeVozača - IDVozača - NazivKomitenta - IDKomitenta'.

Zadatak 7

```
Create TRIGGER Vozac-Komitent
      ON Prevoz
      FOR Update
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDVzn int
      Declare @IDPos int
      Declare @IDKom int
      Declare @IDVoz int

      SET @MyCursor = CURSOR FOR
      Select IDPos, IDVzn
      From inserted

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor
      INTO @IDPos, @IDVzn

      WHILE @@FETCH_STATUS = 0
      BEGIN
            SELECT @IDKom = IDKom
            FROM Posiljka WHERE IDPos = @IDPos

            SELECT @IDVoz = IDVoz
            FROM Voznja WHERE IDVzn = @IDVzn

            IF NOT EXISTS (
               SELECT *
               FROM Vozac_Komitent
               WHERE IDVoz = @IDVoz AND IDKom = @IDKom)
            Begin
               INSERT INTO Vozac_Komitent
                 (IDVoz, Ime, IDKom, Naziv)
               VALUES (
                 @IDVoz,
                 (SELECT Ime FROM Vozac WHERE IDVoz = @IDVoz),
                 @IDKom,
                 (SELECT Naziv FROM Komitent WHERE IDKom = @IDKom)
               )
            End

            FETCH NEXT FROM @MyCursor
            INTO @IDPos, @IDVzn
          END

          CLOSE @MyCursor
          DEALLOCATE @MyCursor
        End
```

Zadatak 7

```
Create TRIGGER Vozac-Komitent2
      ON Vozac
      FOR Update
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDVoz int
      Declare @Ime varchar(20)

      SET @MyCursor = CURSOR FOR
      Select IDVoz, Ime From inserted

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor INTO @IDVoz, @Ime

      WHILE @@FETCH_STATUS = 0
      BEGIN
            UPDATE Vozac_Komitent SET Ime = @Ime
            WHERE IDVoz = @IDVoz

            FETCH NEXT FROM @MyCursor INTO @IDVoz, @Ime
      END

      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```

```
Create TRIGGER Vozac-Komitent3
      ON Komitent
      FOR Update
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDKom int
      Declare @Naziv varchar(20)

      SET @MyCursor = CURSOR FOR
      Select IDKom, Naziv From inserted

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor INTO @IDKom, @Naziv

      WHILE @@FETCH_STATUS = 0
      BEGIN
            UPDATE Vozac_Komitent SET Naziv = @Naziv
            WHERE IDKom = @IDKom

            FETCH NEXT FROM @MyCursor INTO @IDKom, @Naziv
      END

      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```

Zadatak 8

Šema RBP „Rent a car“:

AUTOMOBIL (IDAut, RegBr, BrSedista, PredjenoKm)
VOZAC (IDVoz, Ime, Dozvola)
MESTO (IDMes, Naziv)
KOMITENT (IDKom, Naziv, Adresa)
FIRMA (IDKom, ZiroRacun)
OSOBA (IDKom, JMBG)
CENA (IDCen, DatumVaziOd, CenaKM, CenaSat)
VOZNJA (IDVzn, DatumVremeOd, DatumVremeDo, IDMesOd, IDMesDo, PredjenoKm, IDAut, IDKom)
UCESCE (IDVzn, IDVoz)
RACUN (IDRac, Mesec, DatumIzdavanja, IDKom)
STAVKA (IDSta, Iznos, IDVzn, IDRac)
UPLATA (IDUpl, Mesec, Datum, Iznos, IDRac)
VOZI(IDAut, IDVoz, DatumVremeOd)
JE_U_MESTU (IDAut, IDMes, DatumVremeOd)
KORISTI (IDKom, IDAut, IDMesOd, DatumVremeOd)

Zadatak 8

Pod pretpostavkom da aplikacija pri kraju vožnje ažurira samo tabelu VOZNJA, sastaviti SQL okidač koji ažurira ostale podatke u bazi podataka.

Zadatak 8

```
Create TRIGGER KrajVoznje
      ON Voznja
      FOR Insert
      As
Begin
      Declare @MyCursor CURSOR
      Declare @IDVzn int
      Declare @IDAut int
      Declare @PredjenoKm int
      Declare @IDMesDo int
      Declare @DatumVremeDo datetime
      Declare @IDVoz int

      SET @MyCursor = CURSOR FOR
      Select IDPos, IDVzn, PredjenoKm,
            IdMesDo, DatumVremeDo
      From inserted

      OPEN @MyCursor
      FETCH NEXT FROM @MyCursor
      INTO @IDAut, @IDVzn, @PredjenoKm,
            @IDMesDo, @DatumVremeDo

      WHILE @@FETCH_STATUS = 0
      BEGIN
            SELECT @IDVoz = IDVoz FROM VOZI
            WHERE IDAut = @IDAut

            DELETE FROM VOZI WHERE IDAut = @IDAut

            INSERT INTO UCESCE VALUES (@IDVzn,@IDVoz)

            UPDATE AUTOMOBIL
            SET PredjenoKm = PredjenoKm + @PredjenoKm
            WHERE IDAut = @IDAut

            DELETE FROM KORISTI WHERE IDAut = @IDAut

            INSERT INTO JE_U_MESTU
            VALUES (@IDAut, @IDMesDo, @DatumVremeDo)

            FETCH NEXT FROM @MyCursor
            INTO @IDAut, @IDVzn, @PredjenoKm,
                  @IDMesDo, @DatumVremeDo
      END

      CLOSE @MyCursor
      DEALLOCATE @MyCursor
End
```